IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
Art Unit Not Known
Examiner Not Known

In Re:	Enrique Musoll et al.
Case:	P3824
Serial No.:	NA
Filed:	06/14/2001
Subject:	Method and Apparatus for Allocating and De-allocating Consecutive Blocks of Memory in Background Memory Management

To the Commissioner of Patents and Trademarks
Washington, D.C. 20231

Letter To The Examiner

Dear Sir,

The above-mentioned applicant has participated in the document disclosure program. The specification for the above-mentioned patent application refers to Document Disclosure Number 491,557, entitled "Hardware Algorithm for Allocating and De-allocationg Consecutive Blocks of Memory", which was assigned a filing date of 04/03/2001. Further, please retain a copy of the disclosure document in the present application file.

If there are any fees due beyond any fees paid with the present application and amendment, such fees are authorized to be deducted from deposit account 50-0534.

Respectfully Submitted,

Enrique Musoll et al.

by _____
Donald R. Boys
Reg. No. 35,074

Donald R. Boys
Central Coast Patent Agency, Inc.
P.O. Box 187
Aromas, CA 95004
(831) 726-1457

CCPA Patent Disclosure Form

This form is provided by CCPA in an effort to obtain enabling written disclosure for the purpose of preparing patent cases on behalf of our clients.

Please take the time to fill it out properly as the provision of clear and concise disclosure will speed the process of preparing and filing your case. In addition a signed disclosure document filed with the patent office can effectively remove references anticipating our invention in the prosecution of the case thereby enhancing our chances of obtaining patents. In most cases, verbal disclosure or short e-mail messages are inadequate forms of disclosure and should be avoided.

To fill out the form correctly, follow each set of instructions provided with each heading.

Title of Invention

This section is simply a brief descriptive title of the invention.

Insert title here: ***Hardware algorithm for allocating and de-allocating consecutive blocks of memory***

Inventors

We will need the residence address, mailing address, full legal names and citizenship of each inventor at the time of submission of the disclosure.

Enrique Musoll
Mario Nemirovsky

Related inventions known or authored by you or your company

This section should list any prior patents known to you or patents that you have already filed if the present invention depends on them for successful practice.

[1] PA3814 "Methods and Apparatus for Background Memory Management"

[2] "Fundamental Algorithms", vol. 1, by Donald E. Knuth. Section 2.5 ("Dynamic Storage Allocation").

Background

This section is used to describe "the state of the art" before being improved or enhanced with your invention. It should include a brief summarization of existing technologies if any that the present invention improves upon or replaces, a description of any specific problems with "the way the art is practiced now", and a very brief statement of what is needed to improve or replace the existing art. Include references by U.S. patent number any closely related patents discovered during any prior-art searches

Begin Background here:

A software-managed memory is costly in terms of instructions that it takes to figure out which portions of the memory are free and which are available.

In this patent disclosure we provide a mechanism to do this management in hardware. This goal of this mechanism is to offload a central processing unit from the task of managing the memory, while minimizing the problem of memory fragmentation. This patent disclosure describes in detail the algorithm mentioned in [1].

Description of Invention

This section should explain the basic apparatus and method of practicing your invention according to a preferred state. If certain **apparatus** of the invention is not known in the prior art then indicate so. If a **method** of the present invention is not known in prior art then indicate so. If certain methods and apparatus are known in prior art then they do not have to be greatly detailed. However any new subject matter novel over the prior art should be fully explained and represented by drawings and/or sketches.

Begin description here:

Introduction

The goal of the proposed mechanism is to provide an efficient way to consecutively store packets into a memory with minimal memory fragmentation.
The memory that will be managed by the proposed algorithm is divided into blocks of a fixed size. In XCaliber, the total memory controlled by this hardware scheme is 256KB, and it is divided into 4 blocks of 64KB each. Moreover, each block is divided into smaller sub-blocks, named atomic pages, of 256 bytes each. Therefore, there are 256 atomic pages in a block, and a total of 1024 atomic pages in the hardware controller memory in XCaliber.

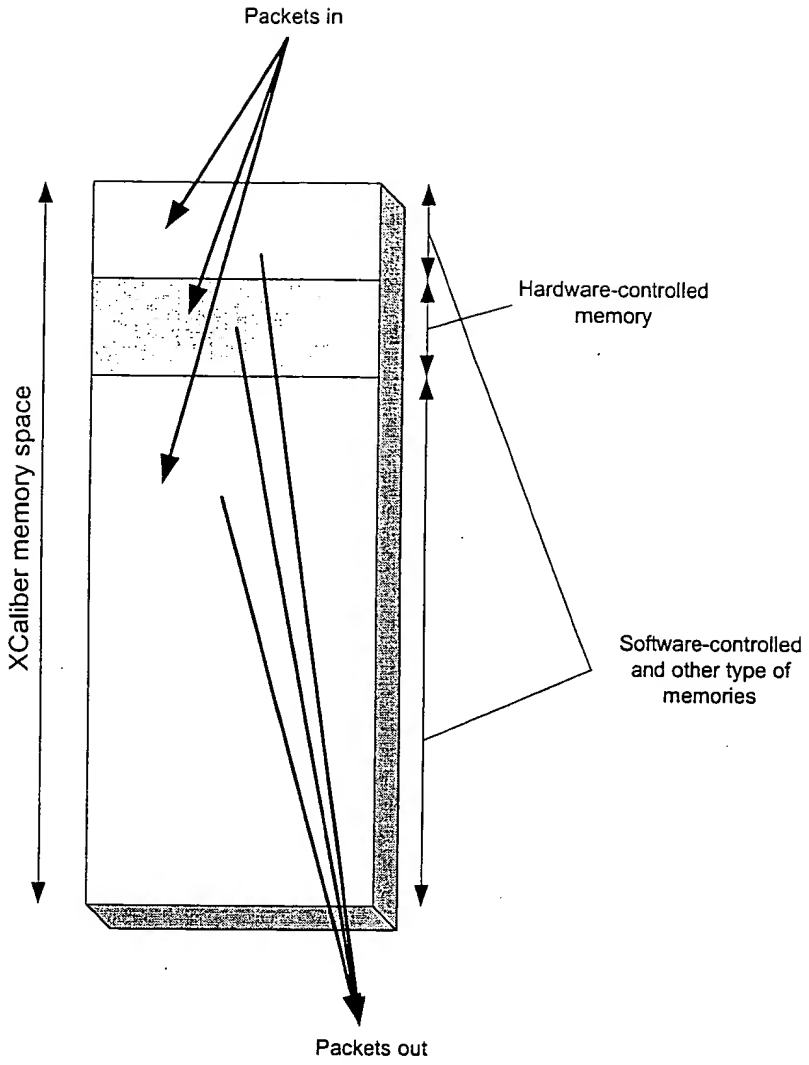


Figure 1: Hardware-controller memory within XCaliber's memory space.

Figure 1 *shows, within the memory space of XCaliber, the portion of memory controlled by the proposed hardware. It is desirable that the incoming packets get stored in this memory so that the central processing unit does not need to perform the costly operations of storing the packet, and likewise, the operations of reading the packet out when the packet has to be sent out of XCaliber.*

There is the possibility that software can control none, one, two, three or all four of these blocks, thus leaving the rest to the control of the hardware algorithm. A configuration flag exists per block for this purpose.

The hardware that implements the algorithm (henceforth named HAL) will not store any incoming packet into a block that has the associated flag asserted.
HAL determines whether the incoming packet fits in any of the blocks. If it does, it determines which of the four blocks will be selected to fit the packet (because the packet may fit in more than one block), and the first and last atomic page that the packet uses in the selected block. These atomic pages are allocated for the incoming packet, i.e. they are marked by HAL as busy. When the packet data stored in an atomic page has been read by HAL when the corresponding packet is sent out of XCaliber, the atomic page is de-allocated and marked as free.
In XCaliber, HAL determines the size of the incoming packet by examining the first two bytes of the packet. In case the packet does not fit into the hardware-controlled memory, HAL will request to another hardware entity within XCaliber to store the incoming packet in a different memory.

Virtual Pages

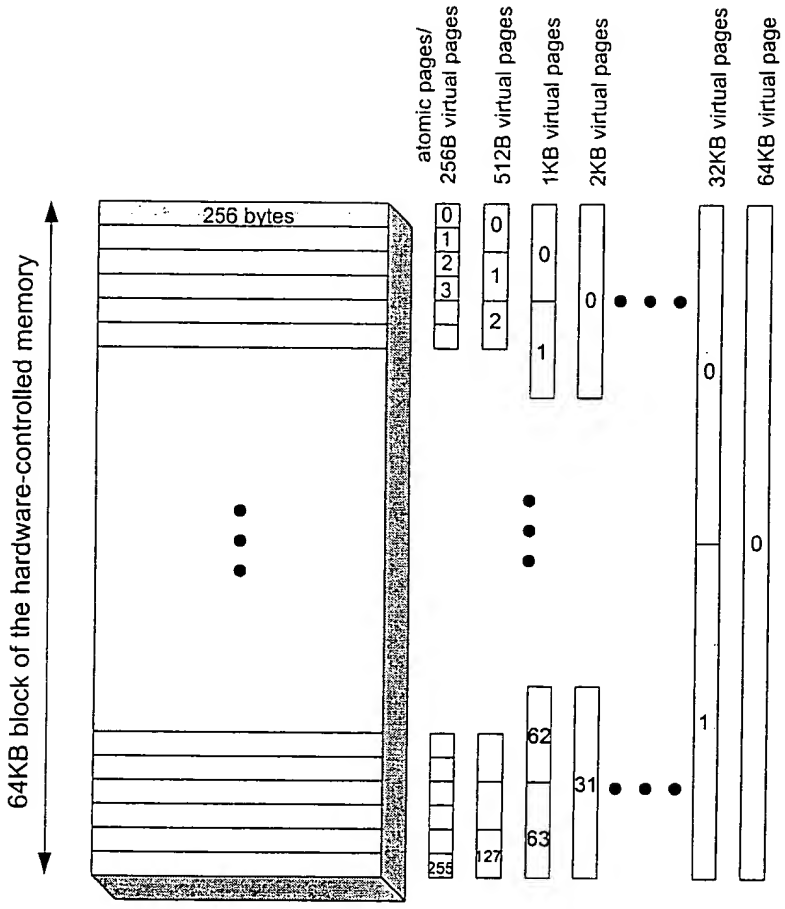


Figure 2: Virtual Pages

HAL introduces the concept of a virtual page. A virtual page is a set of one or more atomic pages. A virtual page is composed of atomic pages or of other virtual pages. Figure 2 *shows the different virtual pages that exist in one of the 64KB blocks of the hardware-controlled memory.*

For each virtual page, a flag is maintained or computed by HAL to figure out whether the virtual page has been allocated or not. Note that this computation can be made if the status of the different atomic pages is known.

Several decisions have been taken to lower the cost of the hardware implementation of the algorithm. These are:

- *The more blocks the memory is divided into, the lower the cost since the number of atomic pages per block is smaller and therefore, the logic associated to search for available atomic pages is also smaller.*
- *The larger the atomic page size, the lower the cost since the number of atomic pages within a block is proportional to the size of the atomic page.*
- *The different number of sizes of virtual pages. The chosen power-of-two increment structure provides a cost-effective way to have virtual pages. This power-of-two increment decision has been already proposed in prior art [2].*

Note that these measures to reduce cost conflict with the goal of minimizing the memory fragmentation.

Algorithm

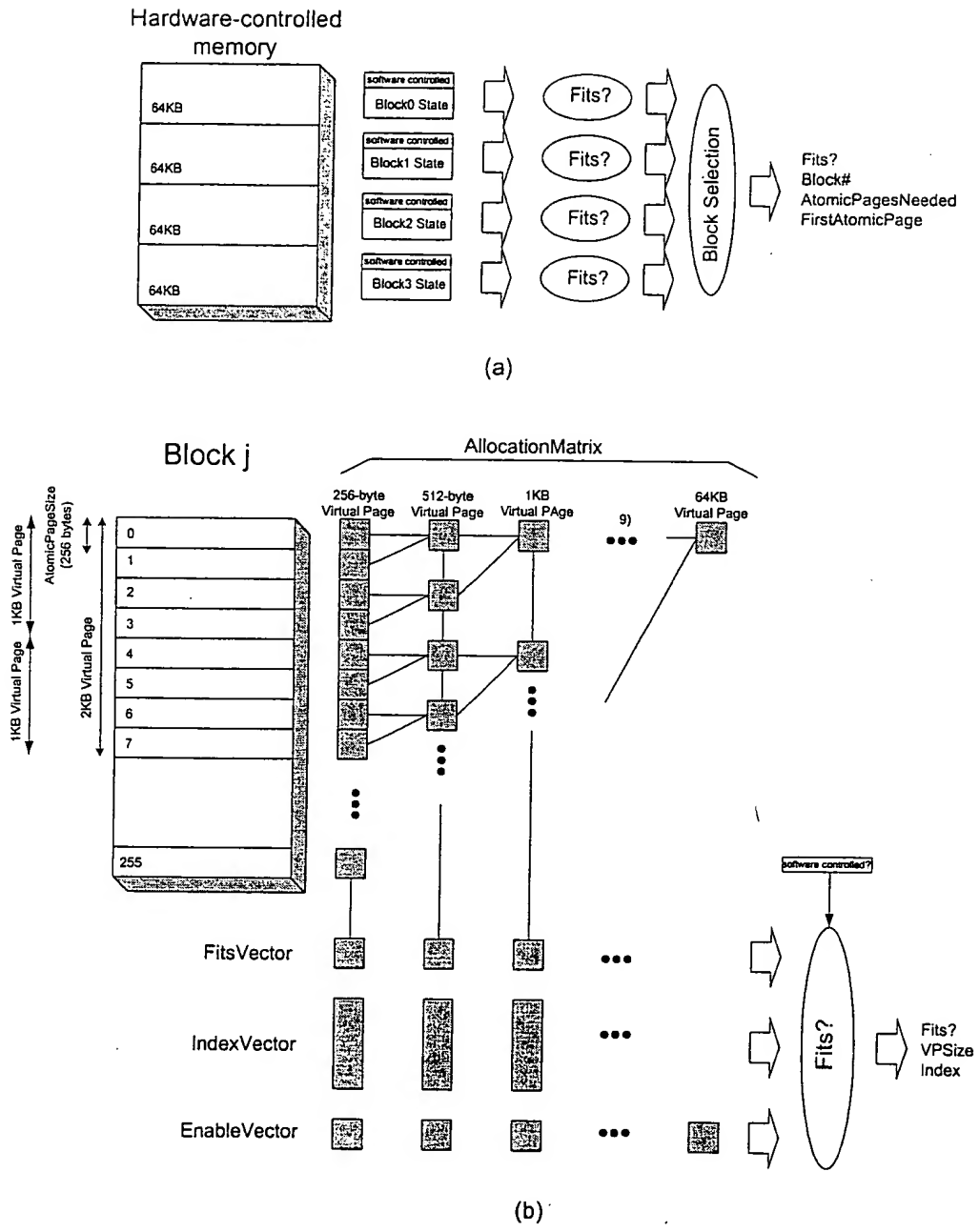


Figure 3: Allocation algorithm implementation.

The following is a particular implementation of the algorithm that requires two cycles. Figure 3 *illustrates this implementation.*

In the first cycle the determination of whether the packet fits into the memory is made. In the subsequent cycle, the atomic pages needed to store the packet are allocated and the new state (allocated/de-allocated) of the virtual pages is computed.

The fits determination logic is performed in parallel for all the four blocks (see Figure 3*(a)). The state of each of the virtual pages per block, called AllocationMatrix[][] (see* Figure 3*(b)), is recomputed when one or more atomic pages are allocated or de-allocated, and it is an input to the determination logic. The FitsVector[] and IndexVector[] contain information computed from the AllocationMatrix[][]:*

- *AllocationMatrix[VPSize][VPIndex]: indicates whether virtual page number VPIndex of size VPSize is already allocated or not.*
- *FitsVector[VPSize]: indicates whether the block has at least one non-allocated virtual page of size VPSize.*
- *IndexVector[VPSize]: if FitsVector[VPSize] is asserted, this vector contains the index of a non-allocated virtual page of size VPSize.*

Software determines the virtual page sizes that are enabled for each block. The EnableVector[VPSize] contains this information.

Note that the AllocationMatrix[][], FitsVector[] and IndexVector[] are in the undefined state if the block is managed by software.

The algorithm for the fits determination logic (for a packet of size s bytes) is:

1) *Fits logic: Check, for each of the blocks, whether the packet fits in or not. If it fits, remember the virtual page size and the number of the first virtual page of that size.*

```
For All Block j Do (can be done in parallel):
        Fits[j] = (s <= VPSize) AND FitsVector[VPSize] AND
                        Not SoftwareOwned
                  where VPSize is the smallest possible page size.
        If (Fits[j])
                VPIndex[j] = IndexVector[VPSize]
                MinVPS[j] = VPSize
        Else
                MinVPS[j] = <Infinit>
```

2) *Block selection: The blocks with the smallest virtual page (enabled or not) that is able to fit the packet in are candidates. The block with the smallest enabled virtual page is selected.*

```
If Fits[j] = FALSE for all j Then
        <Packet does not fit in hardware-controlled memory>
Else
        C = set of blocks with smallest MinVPS AND Fits[MinVPS]
        B = block# in C with the smallest enabled virtual page
                (if more than one exists, pick the smallest block number)
        If one or more blocks in C have virtual pages enabled Then
                Index = VPIndex[B]
                VPSize = MinVPS[B]
                NumAPs = ceil(S/256)
```

*packetPage = (B*64KB + Index*VPSize) >> 8*

Else

<Packet does not fit in hardware-controller memory>

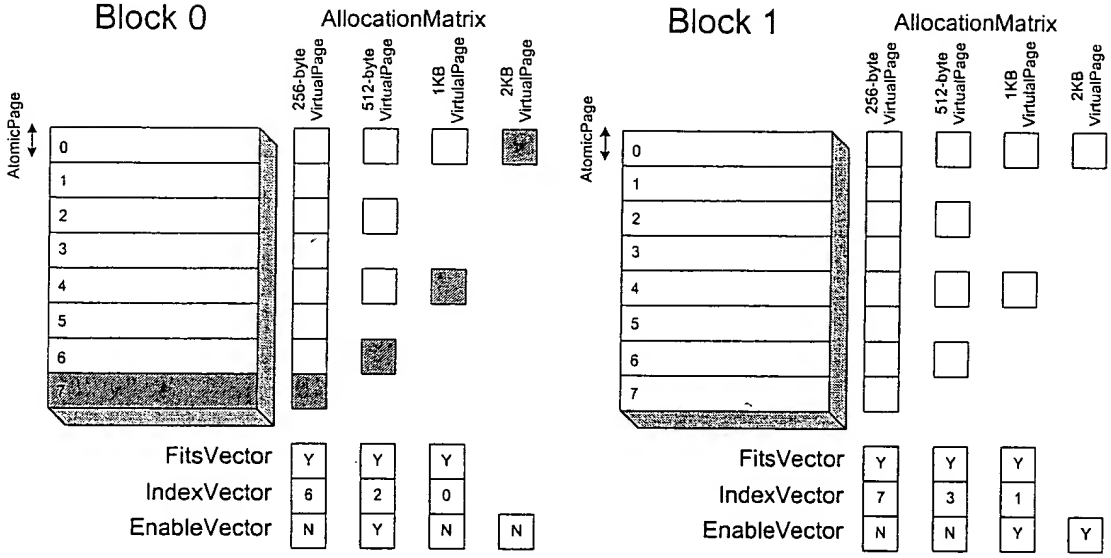
The packetPage is the atomic page number of the first atomic page that the packet will occupy in the hardware-controlled memory. The packetPage is then an offset within the hardware-controlled memory and it can be used by software to access all the data of the packet stored in that memory.

The number of atomic pages needed to store the packet is calculated (NumAPs) and the corresponding atomic pages are allocated. The allocation of the atomic pages for the selected block (B) is determined as follows:

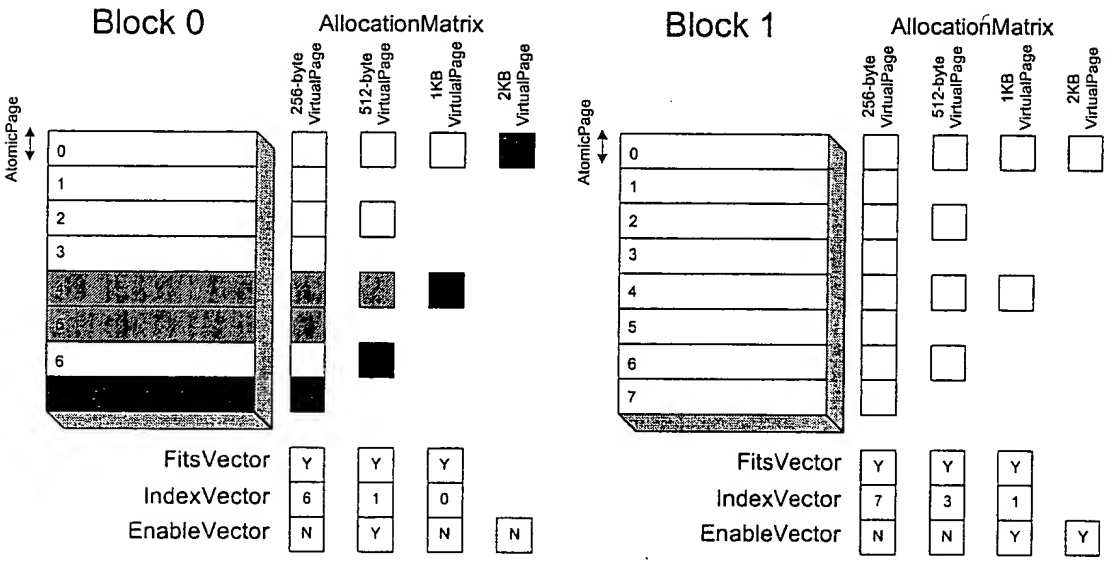
- *The allocation status of the atomic pages in AllocationMatrix[Apsize][j..k], j being the first atomic page and k the last one (k-j+1 = NumAPs), are set to allocated.*
- *The allocation status of the virtual pages in AllocationMatrix[r][s] is updated following the mesh structure shown in* Figure 3*(b): a 2^{k+1}-byte virtual page is allocated if any of the two 2^k-byte virtual pages that it is composed of is allocated.*

- Two blocks, 2KB each
- AtomicPageSize of 256 bytes
- Sequence of packet sizes: 256, 512, 1K, 512, 2K (does not fit)

Packet of 256 bytes



Packet of 512 bytes



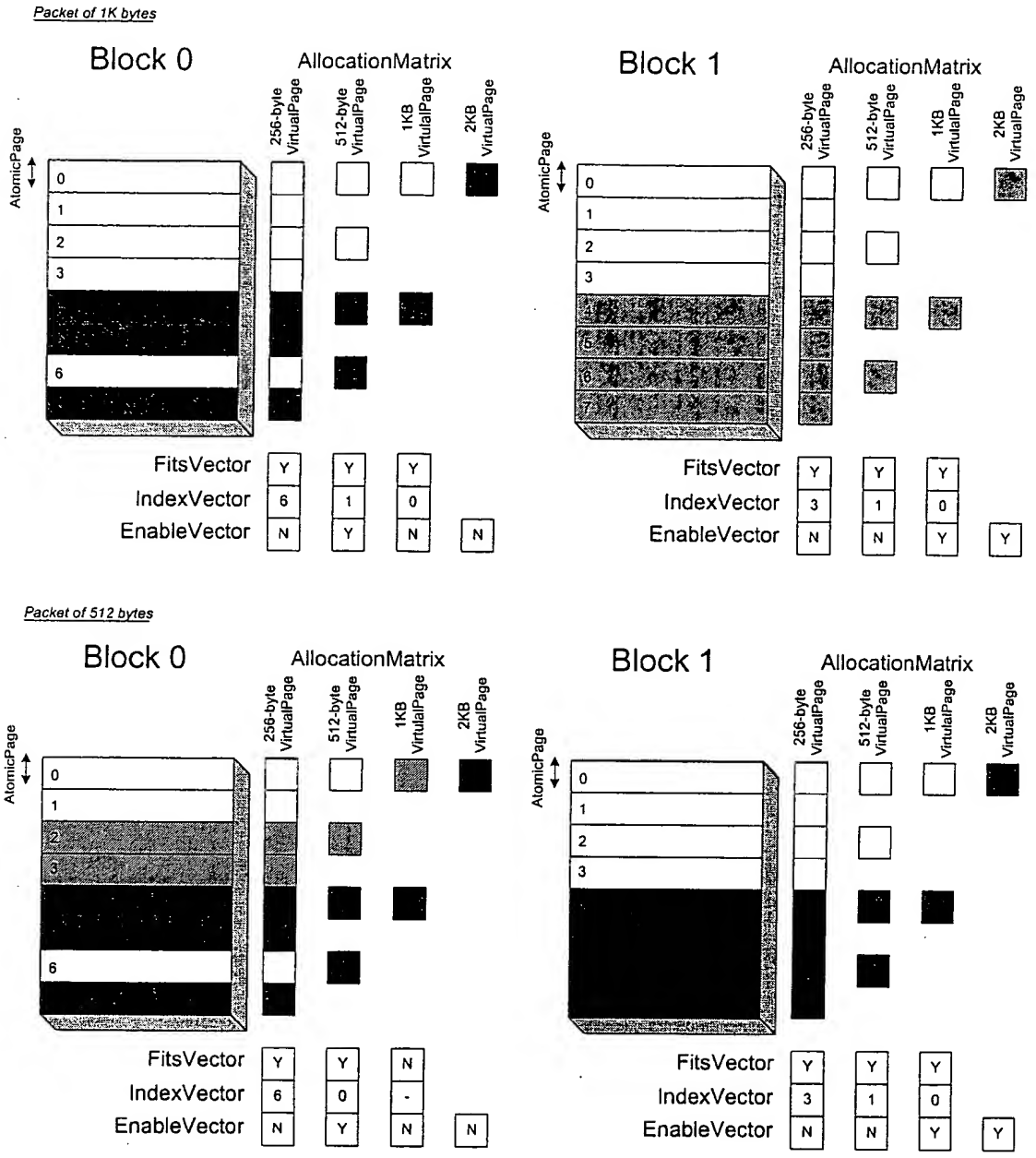


Figure 4: Example of the allocation algorithm.

Figure 4 *shows an example of how atomic (and virtual) pages are allocated. For simplicity, the example assumes two blocks of 2KB each, and both blocks are not software controlled. In the figure, shadowed and stripped areas represent allocated virtual pages; stripped pages correspond to the pages allocated in the current cycle. The example illustrates how the pages become allocated for a sequence of packet size of 256 bytes, 512 bytes, 1KB and 2KB. Note that after this sequence, a 2-KB packet, for example, can not fit in this simplified example.*

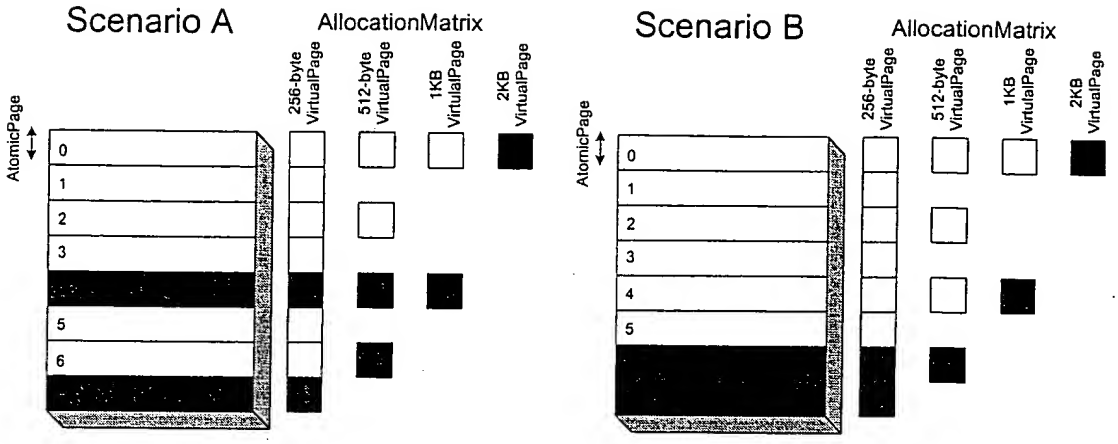


Figure 5: Index selection.

When the FitsVector[VPSize] is asserted, the IndexVector[VPSize] contains the largest non-allocated virtual page index for virtual page size VPSize. The reason for choosing the largest index is that the memory space is better utilized. This is shown in Figure 5, *where two 256-byte packets are stored in a block. In scenario A, the 256-byte virtual page is randomly chosen; in scenario B, the largest index is always chosen. The block in scenario A only has two 512-byte virtual pages free, whereas the block in scenario B allows three. Both allow the same number of 256-byte (or less) packets because 256-byte is the smallest allocation unit. Note that the same effect is obtained by choosing the smallest virtual page index number all the time.*

Please have all inventors sign the disclosure and mail a hard copy to CCPA for participation in the document disclosure program. Also e-mail to Mark Boys markboys@centralcoastpatent.com and CC Don Boys rexboys@centralcoastpatent.com

Your cooperation in the filling and return of this form will expedite the processing of your application and increase our chances of obtaining a patent for your invention.

Mark Boys, CCPA